CLAIMS

What is claimed is:

1. A method of increasing the execution speed of invoking and returning from a Method while reducing the supporting memory footprint, the method comprising:

establishing an activation stack frame template with a set of criteria;

determining whether the Method conforms to the criteria of the stack frame template;

creating a fixed size activation frame regardless of the Method's exact stack requirements, based on the set of criteria of the activation stack frame template if the Method conforms to the set of criteria of the activation stack frame template;

creating an activation frame to match the Method's exact stack requirements if the Method does not conform to the set of criteria of the activation stack frame template; and

associating a Method access structure with the Method such that the Method access structure is contiguous with the code of the Method.

2. A method as claimed in claim 1, wherein the set of criteria includes the number of parameter words, the total number of local words, and the number or words of evaluation stack.

3. A method as claimed in claim 1, wherein creating an activation frame for the Method based on the set of criteria of the activation stack frame template includes creating a local variable portion, an evaluation stack, and a fixed size frame linkage structure.

4. A method as claimed in claim 1, further comprising associating the Method access structure with a pointer and defining the pointer such that it is an indicator of where code for implementing a Method resides and an indicator for the Method itself.

5. A method as claimed in claim 1, wherein the Method access structure is variably sized.

6. A method of increasing the execution speed of invoking a plurality of Methods in an execution device, the plurality of Methods associated with one or more classes, the method comprising:

establishing an activation stack frame template with a set of criteria;

for each one of the plurality of Methods,

determining whether the one Method conforms to the criteria of the stack frame template;

creating a fixed size activation frame regardless of the one Method's exact stack requirements, based on the set of criteria of the activation stack frame template if the one Method conforms to the set of criteria of the activation stack frame template;

creating an activation frame to match the one Method's exact stack requirements if the one Method does not conform to the set of criteria of the activation stack frame template; and

associating a Method access structure with the Method such that the Method access structure is contiguous with the code of the Method;

creating a Method routing structure for each class; and

rewriting invocation bytecodes to a form that includes an indication of the Method routing structure.

7. A method as claimed in claim 6, wherein the set of criteria includes the number of parameter words, the total number of local words, and the number or words of evaluation stack.

8. A method as claimed in claim 6, wherein creating an activation frame for the Method based on the set of criteria of the activation stack frame template includes creating a local variable portion, an evaluation stack, and a fixed size frame linkage structure.

9. A method as claimed in claim 6, further comprising associating each Method access structure with a pointer and defining the pointer such that it is an indicator of where code for implementing a Method resides and an indicator for the Method itself.

10. A method as claimed in claim 6, further comprising maintaining Method access structures associated with dynamically compiled code in an area of memory separate from Method access structures associated with bytecode.

11. A method as claimed in claim 6, further comprising creating the Method routing structure such that it has one or more misaligned pointers.

12. A method as claimed in claim 11, wherein the misaligned pointers are used to denote processor executable Method access structures and one or more aligned pointers are used to denote processor non-executable Method access structures.

13. A method as claimed in claim 6, wherein each Method access structure is variably sized.

14. An execution system for increasing the execution speed of invoking Methods of one or more classes, the system comprising:

memory; and

a virtual machine operable to access the memory, to create a representation of at least one of the Methods based on an activation stack frame template with a set of criteria, to create a representation of at least one of the Methods based on exact stack requirements, to spatially associate a Method access structure contiguous to the representation of each of the Methods, and to create a Method routing structure for each of the one or more classes in the memory.

15. An execution system as claimed in claim 14, wherein the set of criteria includes the number of parameter words, the total number of local words, and the number or words of evaluation stack.

16. An execution system as claimed in claim 14, wherein the activation frame template includes a local variable portion, an evaluation stack, and a fixed size frame linkage structure.

17. An execution system as claimed in claim 14, wherein the virtual machine is operable to associate a pointer with each Method access structure, the pointer defined such that it is an indicator of where code for implementing a Method resides and an indicator for the Method itself.

18. An execution system as claimed in claim 14, wherein the virtual machine is operable to maintain Method access structures associated with dynamically compiled code in an area of memory separate from Method access structures associated with bytecode.

19. An execution system as claimed in claim 14, wherein the Method routing structure includes one or more misaligned pointers to denote processor executable Method access structures.

20. An execution system as claimed in claim 14, wherein the virtual machine is operable to spatially associate the Method access structure immediately preceding the representation of each of the Methods.